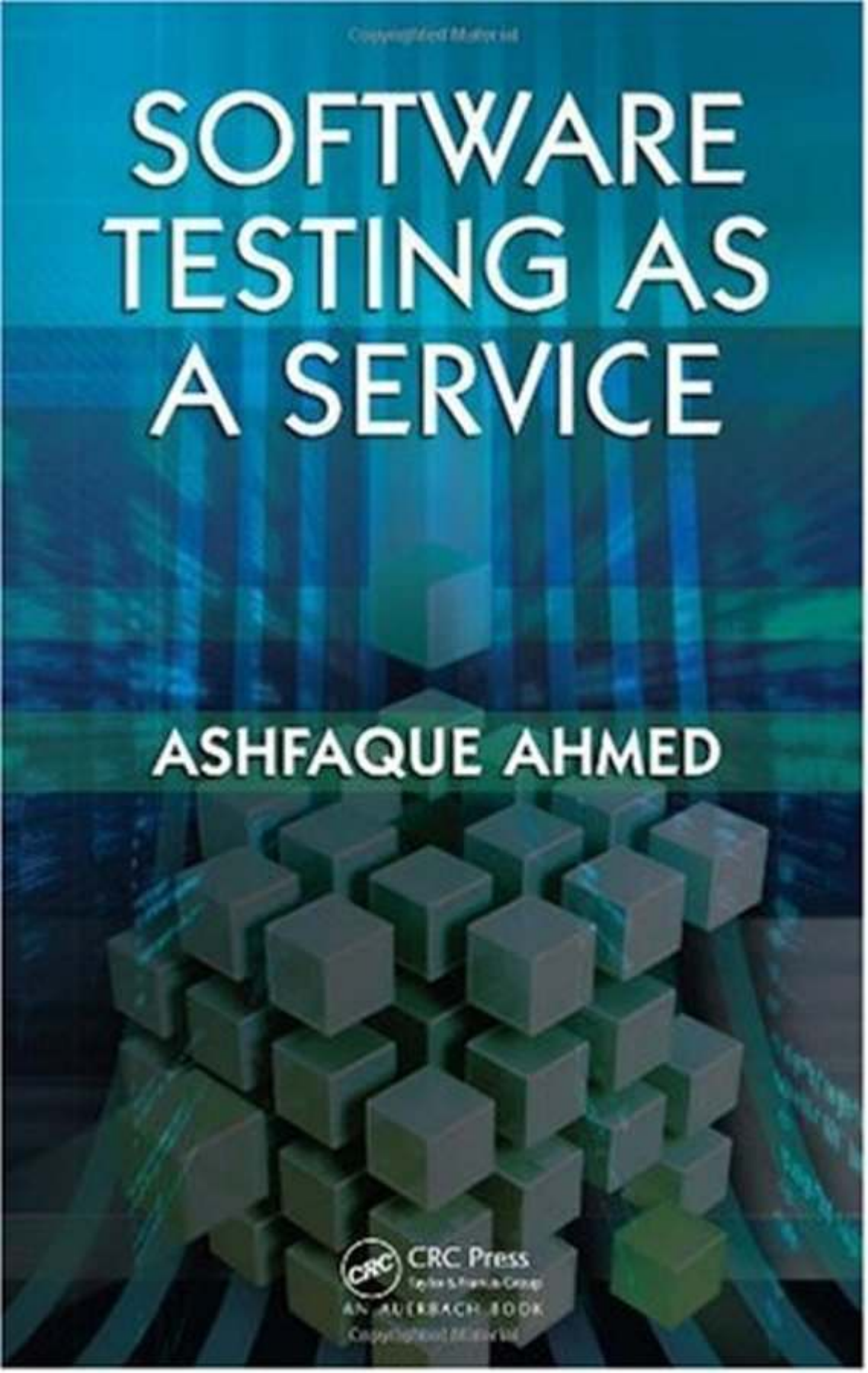
SOFTWARE TESTING AS A SERVICE
ASHFAQUE AHMED
CRC Press
AN AUERBACH BOOK

SOFTWARE TESTING AS A SERVICE

ASHFAQUE AHMED


CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

SOFTWARE TESTING AS A SERVICE

Other Auerbach Publications in Software Development, Software Engineering, and Project Management

Optimizing Human Capital with a Strategic Project Office
J. Kent Crawford and Jeannette Cabanis-Brewin
978-0-8493-5410-6

The Business Value of IT
Michael D.S. Harris, David Herron, and Sasia Iwanicki
978-1-4200-6474-2

Best Practices in Business Technology Management
Stephen J. Andriole
978-1-4200-633-2

Effective Software Maintenance and Evolution
Stanislaw Jarzabek
978-0-8493-3592-1

Interpreting the CMMI®, Second Edition
Margaret Kulpa and Kent Johnson
978-1-4200-6502-2

Global Engineering Project Management
M. Kemal Atesmen
978-1-4200-7393-5

Manage Software Testing
Peter Farrell-Vinay
978-0-8493-9383-9

Programming Languages for Business Problem Solving
Shouhong Wang and Hai Wang
978-1-4200-6264-9

Patterns for Performance and Operability
Chris Ford, Ido Gileadi, Sanjiv Purba, and Mike Moerman
978-1-4200-5334-0

The Handbook of Mobile Middleware
Paolo Bellavista and Antonio Corradi
978-0-8493-3833-5

Managing Global Development Risk
James M. Hussey and Steven E. Hall
978-1-4200-5520-7

Implementing Electronic Document and Record Management Systems
Azad Adam
978-0-8493-8059-4

Leading IT Projects: The IT Managers Guide
Jessica Keyes
978-1-4200-7082-8

A Standard for Enterprise Project Management
Michael S. Zambruski
978-1-4200-7245-7

The Art of Software Modeling
Benjamin A. Lieberman
978-1-4200-4462-1

The Complete Project Management Office Handbook, Second Edition
Gerard M. Hill
978-1-4200-4680-9

Building Software: A Practitioner's Guide
Nikhilesh Krishanmurthy and Amitabh Saran
978-0-8493-7303-9

Software Engineering Foundations
Yingxu Wang
978-0-8493-1931-0

Service Oriented Enterprises
Setrag Knoshafian
978-0-8493-5360-4

Effective Communications for Project Management
Ralph L. Kliem
978-1-4200-6246-5

Software Testing and Continuous Quality Improvement, Third Edition
William E. Lewis
978-1-4200-8073-3

The ROI from Software Quality
Khaled El Emam
978-0-8493-3298-2

Software Sizing, Estimation, and Risk Management
Daniel D. Galorath and Michael W. Evans
978-0-8493-3593-8

Six Sigma Software Development, Second Edition
Christine B. Tayntor
978-1-4200-4462-3

Elements of Compiler Design
Alexander Meduna
978-1-4200-6323-3

Determining Project Requirements
Hans Jonasson
978-1-4200-4502-4

Practical Guide to Project Planning
Ricardo Viana Vargas
978-1-4200-4504-8

Service-Oriented Architecture
James P. Lawler and H. Howell-Barber
978-1-4200-4500-0

Building a Project Work Breakdown Structure
Dennis P. Miller
978-1-4200-6969-3

Building and Maintaining a Data Warehouse
Fon Silvers
978-1-4200-6462-9

AUERBACH PUBLICATIONS
www.auerbach-publications.com
To Order Call: 1-800-272-7737 • Fax: 1-800-374-3401
E-mail: orders@crcpress.com

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-9956-0 (Softcover)

Library of Congress Cataloging-in-Publication Data

Ahmed, Ashfaque.
Software testing as a service / Ashfaque Ahmed. -- 1st ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4200-9956-0 (alk. paper)
1. Computer software--Testing. 2. Computer software industry. I. Title.

QA76.76.T48A53 2009
005.1'4--dc22 2009009590

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface

Contrary to the title of the book, *Software Testing as a Service*, we can think of software testing as a manufacturing activity and not as a service. Yet, by doing so we can commoditize the software testing service. This can have tremendous impact on schedules, quality, and cost, and is now possible, because now many service providers can get many software testing projects to work on at the same time. It would not be possible if any service provider does not get many projects to work on simultaneously. The software services provider can set up a centralized organization that can handle all existing and incoming test projects and deliver them using shared resources. Each project is like a manufactured item that is in queue and then, when its turn comes, gets processed using the resources and then delivered. In fact parts of a project (modules, test cases, etc.) can be considered as inventories that need to be queued and then processed. Each such inventory will have three states (i.e., queued, work-in-process, and finished.) As in manufacturing, we can have production lines and processing departments.

Doesn't this look like a utopian idea? Well, it is just short of happening in reality. Many offshore outsourcing service providers have reached the stage of consolidating test projects and using shared resources to deliver these projects and are in fact delivering the goods, saving millions of dollars for their customers. Using this new methodology, we can do and save even more for our customers and yet still deliver with unmatched quality and with a record reduction in schedule.

I had been thinking of writing a book on this subject for some time, as I felt there was a need to take a step forward and think of a way to achieve something that is not possible via traditional functioning. To express this, I have coined a new phrase: *mass servicing*, akin to *mass manufacturing*. It is my belief that we can achieve something similar to what mass manufacturing achieved for the manufacturing sector and for society at large and in the process, we can truly commoditize software testing. Chapter 12 of the book discusses this concept of software testing as a manufacturing activity in detail. Chapter 11 discusses the benefits of offshoring test projects, including even and consistent quality across projects, consolidation of projects, and sharing of resources. Chapter 13 discusses the relationships

among quality, standards, and software testing. This is most important because many people get confused regarding the difference between software quality and software testing. This chapter discusses and explains the differences and the relationships between the two. Other chapters discuss project planning, risk management, customer expectation management, project reporting, project execution, and some other aspects of software testing. There are checklists provided in Chapters 3 through 8 that will be very useful for test managers.

Readers of this book will benefit by gaining new ideas about software testing from the perspective of a customer who outsources his project to an outsourcer. However, the book deals more with addressing challenges from the outsourcing partner's perspective and that the software test project is being executed from an offshore location. That is why the title of the book is *Software Testing as a Service*.

Since I have a strong grounding in manufacturing, I have tried to explain many software testing concepts from the manufacturing perspective. This is a new approach for a book written on the subject of software testing management.

Conventions in this book are per the standards specified in the Capability Maturity Model for Integration (CMMI). The time has come for software development and software maintenance and support (after deployment services) to be governed not by separate processes, but rather under a single umbrella. This is the reason all illustrations and concepts in this book follow CMMI standards.

Great effort has been made to explain concepts in an easy-to-understand style using layman's language. Consequently, even beginners will be able to understand these topics. Examples and case studies have also been provided wherever possible to make it easy to grasp these concepts.

A great effort has also been made to discuss only current and relevant topics. This will help readers to get current and useful information.

I strongly believe that the readers of this book will benefit greatly from the information presented within. This is my sole purpose in writing this book. I hope the book will succeed in its intended goal.

Ashfaque Ahmed

Acknowledgments

This book is dedicated to my mother, Aesha, and the kids in the family, Sofia, Arisha, Shija, Ashi, and Jasim. I have always gotten inspiration from my mother, who has supported me in everything I have done in my life. Likewise, the kids have always shown me how to be happy even when life looks to be in dire straits. They are the chirpy little people in my life. I hope someday they will become better writers than I am. I am also indebted to my brother Javed, who has supported me in everything I have done in my life. His contribution has been tremendous and has included helping me with the cover design, images, and reviews for this book. My youngest brother, Aslam, has also helped me prepare material for this book.

In my professional life (spanning more than 20 years now) I have come across some wonderful people from whom I learned how to be successful. I am thankful to all of them. In particular, I am especially thankful to Gary Hahn of One Network Enterprises, who has been a wonderful manager and a caring human being, and has also contributed by reviewing this book. I am also thankful to my colleague Dinesh Salvi, who reviewed the book, as well as all my colleagues at Entercoms Software Private Limited with whom I shared unforgettable moments while working on our assignments.

In my personal life I have met influential people whose perspective helped me to understand many aspects of my life better. One such person is Nandu Phadke, a distinguished lawyer and a person with great integrity.

I am also thankful to my publisher, Auerbach Publications, who gave me the opportunity to write this book.

About the Author

Ashfaque Ahmed is a consultant for software testing and supply chain management. He has over 20 years of experience in the software industry and has served clients with innovative solutions that have saved millions of dollars. He has worked with both midsize and large multinational customers on engagements involving implementing and testing enterprise software applications. The industries served included manufacturing, retail, distribution, and transportation. He also has worked on product development for a midsize enterprise software vendor.

Over the years Mr. Ahmed has written many research papers that were published by Technology Evaluation Centers Inc. (http://www. technologyevaluation.com).

Recently Ahmed started SCM Consulting (http://www.scmconsultingonline.com), a portal that provides valuable information to students and software professionals on areas covering software testing, implementation, development, and selection.

Ahmed holds a bachelor's degree in engineering and an MBA in information systems and is also a Microsoft Certified Professional.

Chapter 1

Introduction to Software Testing Management

Believe me, software testing management is one of the most difficult tasks out there. There are many reasons for this. First, a software product is not a tangible thing that can be measured, physically felt, or sampled. So it is difficult to test a software product. Second, software testing is still not considered a recognized trade and so finding professionally qualified people for the testing job is difficult. Third, unlike well-defined and standardized processes for product design, product development, quality control, and so on, which exist for any product development activity, similar standardized processes have yet to be defined for software testing. Fourth, tools for automation of software testing activities are still in their nascent stage, and it will take time to have sophisticated automation tools available for software testing activities. Fifth, effort estimation techniques for software testing activities are still being evolved, and currently effort estimation is done mostly on an ad hoc basis.

Yet, the importance of software testing is so immense! Any failure of the software product or application can cause damages to the tune of millions of dollars for any company. Even if the software defect is not so big, the support cost can run in the thousands of dollars over the life of the software product.

To better understand software testing management, let us first try to understand what is a defect in a product, how it affects a user, what the user feels when he finds a defect in a product after buying and using it, how to prevent defects, and finally how to identify and remove defects in the physical world. Then from there we can go to software engineering and software testing. From there we can move on to software testing management aspects.

1.1 Product Defect: A Case Study

Just the other day I bought a semi-automatic washing machine. It had a 2-year warranty on it, and the deal looked fine. So I bought it. My wife started using it, and everything looked fine till one day. That day my wife phoned me saying the washing machine is not working. I was in the midst of an office meeting. I was deeply involved in my task at hand and I got disturbed. Now my wife is very blunt when it comes to talking on the phone and when she says something to me, it is a command and not a request. I must do something immediately or else I face the risk of being called an incompetent person. Even if I am busy with something important, I must obey. Cursing my wife and the washing machine company, I told my wife to find the customer service phone number of the washing machine company from the product brochure that came with the washing machine and get it fixed. I also told my wife not to disturb me, as she can handle these things herself. When I came back home that day, I learned that the mechanic had found something stuck in the drainage system of the washing machine and the mechanic cleaned and fixed it. A few days later, my wife phoned me, saying that the washing machine again has some problem. I told my wife not to bother me, as she has the customer service phone number of the washing machine company and she can phone them directly. So my wife phoned the washing machine company and got the machine fixed. This happened many times over the next few months. Finally I decided to confront the service engineer myself and tried to get an explanation as to why it was happening. I found out that that particular brand of washing machine had a design problem and this problem was happening with most of the washing machines of that brand. I talked to customer service and explained my problem. Even after a lot of heated exchanges I was not able to find a solution from the company. Then I thought I should look inside the washing machine and find out the problem myself. So here I was disassembling the washing machine and trying to find out the root cause of the problem. I found out that there was a valve (to stop or start draining of water from the washing machine) connected to the drainage system of the machine and the valve used to activate when a lever attached to the machine was pushed. Dirt coming out of the clothes was getting deposited at the valve and clogging it. So the valve was not closing properly and water was getting drained even when the valve was in the closed position. Water should not come out when clothes are being washed. When washing is complete, you can open the valve to let the dirty water out of the washing machine by pressing the valve lever. I cleaned the valve and then tested it by filling water in the washing machine. I inspected the valve and the lever to find out the problem. I found out that even when the valve was clean the lever was not properly closing the valve and so the valve was getting only half closed. This was the root cause of the problem. I also found out that there was no option to adjust the lever so that it should be able to close the valve completely when operated. So far I was able to trace the root cause of the problem. Now I had to fix it. After thinking much, I decided to bend a part of the lever so that the length of the lever would get

changed and so it can push the valve more and close it more. After some trial and error I was able to make the lever pull the valve perfectly. So the valve was getting opened properly now, but still the valve was not getting closed properly. To this day this problem could not be fixed even after many attempts by service engineers.

1.2 Case Analysis

What is the moral of the story? A small defect in a part of a product can dent confidence of the customers. I for sure will never buy any products from the manufacturer who manufactured that godforsaken washing machine.

This shows the importance of preventing defects in the first place and, if any defect occurs, finding it at the manufacturing site itself and removing that defect so that the defect is not passed to end users or customers.

Preventing as many defects as possible in the product and then trapping defects if any occur at the manufacturing site itself is very important. Right from product conception to product design to production, a process should be followed which will ensure that the product is as close to defect-free as possible. This means that when a machine prototype is passed to the machine design team, the machine prototyping team should ensure that no defects are introduced in the prototyping. When machines made from this design are then installed at the shop floor, the factory should ensure that no defects are passed to the product being made on this machine due to faulty material handling or bad machine design. These measures will ensure defect-free products through a thorough implementation of quality assurance processes. Similarly the product that will be produced using this machine should also go through the same quality assurance process at each stage of product conception to product design to product production. And at each stage in the manufacturing process the quality testing department should ensure that any defects which may occur in the product are trapped and faulty products are either reworked or rejected.

In the physical world preventing and removing defects is very much possible to the extent of having products 99.99999% defect-free. It is possible because a number of factors that cause defects can be determined and then removed from design or during the manufacturing process. Nowadays product development has matured so much that in most of the industries defects due to faulty conception and design are nonexistent. They have also mastered manufacturing processes so that they can achieve defect-free products to the tune of 99.99999%. They have feverishly implemented six sigma programs, ISO standards, lean manufacturing principles, quality circle programs, and so on. So the first principle is that defects should not be introduced in the products. The second principle is that if some defects enter at any stage, then they should be trapped before the product enters into the next stage in the production cycle. One good example of this trend is the introduction of Toyota's manufacturing system. At Toyota Motors the quality processes are so

strong that their cars are virtually defect-free. This has helped Toyota tremendously in capturing market share and reducing their manufacturing costs. The Toyota manufacturing model has become a de facto standard in the automobile industry, and subsequently other industries have followed their example.

1.3 Return on Investment

From another point of view, preventing defects as early as possible presents a compelling ROI perspective. Say cost of finding and fixing a defect at prototype stage is US$40. This defect is not found and is passed on to the design stage, where it is trapped and fixed. In that case the cost of fixing it would be at least fivefold as the defective prototype has gone into design and so design is also flawed and needs to be fixed as well. At the same time this one prototype defect introduces five design defects. After the production stage this cost will become at least 25-fold from the prototype, as now either the product prototype along with the machine design will also have to be fixed or these 25 defects after the design stage have to be fixed. Finally, when it comes to customer site, the cost will be a staggering 125 times that of the prototype stage, as now support cost will also get involved. So the cost of fixing the defect will be US$5000 ($40 × 125). So you see it definitely makes sense to fix the defect as early as possible. Of course, time and cost constraints put a limit on to what extent defects should be traced and fixed. But continuous improvement techniques help to reduce defect injection and improve defect prevention over a period of time once the techniques are introduced. This is what CMM (Capability Maturity Model) talks about.

Coming to a software product, suppose for each quarterly release on average 500 must-fix defects are found. The cost of fixing each defect at build phase is at $100; at testing phase, $500; and at customer end, $2500. Now suppose the software vendor has no testers. Industry standard is that 25% of errors are detected by developers, 50% on average by skilled testers, and the remaining 25% by end users. In our case, because no testers are involved, 75% of total defects are detected and fixed at customer site after 25% of defects are detected and fixed by developers. Total cost of fixing defects comes to $887,500 ($12,500 + $875,000).

Now suppose we employed three software testers at $50 per hour. So their total salary for the quarter comes to $108,000. In this scenario the cost of fixing defects comes to $495,500 ($12,500 + $108,000 + $375,000).

Compared to the previous scenario we can see that we are saving $392,000. This is a saving of more than 44% over the case when no testers are deployed. So it can be clearly seen that there is a great return on investment (ROI). The ROI improves further when the testing function is outsourced and even further when it is offshored.

In Chapter 11, we introduce a new concept of commoditizing software testing service. If this concept can be devised, refined, and finally implemented, it can provide tremendous cost savings to customers.

1.4 Causes of Defects in Software

But why does the software industry still find it difficult to produce defect-free software? Well, there are many reasons for it. Software does not have a physical form that can be seen, and defects can be found during conception, design, or build. The factors that can induce defects in the software products are too numerous to list here. For instance, all the internal and external parts of the software are made up of small parts of software or, in other words, a set of instructions. Internal defects are bound to exist in some of these thousands of sets of instructions inside a big software product so that at the complete software level the number of defects could run in the thousands. There is one more source of defects apart from internal defects. Software is made up of modules, and modules are made up of submodules. These modules and submodules are integrated with each other through their interfaces. Even if the data gets passed through these interfaces during transaction execution, it is not the right data passing through these interfaces. For example, suppose there is a module for product master list and product names are being fetched using this module from other modules. This is done through a mechanism of product code lists that are matched in both modules (product master module and the other module that wants to display product names). Due to the mismatching of product list, it is possible that wrong product names are being displayed. Thus we have a genuine functional error here.

Now let us see a list of causes that introduce defects in software:

- **Miscommunication or no communication**—Software specification or requirements are not understood by the project team due to miscommunication or no communication among the project team.
- **Software complexity**—Software applications are getting more complex due to an increase in complexity in requirements. This complexity leads to more defects creeping into the software application. Proliferation of many kinds of software applications with diverse technology backbones also leads to complexity of software applications. Multitiered applications, client-server and distributed applications, data communications, enormous relational databases, and the sheer size of applications have all contributed to the exponential growth in software/system complexity.
- **Programming errors**—If design specifications are not understood by programmers or if programmers make mistakes in programming structures or implement any design wrongly by mistake, then defects will be introduced in the software.

- **Changing requirements**—This is one of the most important sources of defects in software. A change in requirements upsets the whole project, from design to build to testing to deployment. In extreme situations, it may lead to the throwing away of all already designed and built software. In other cases, it may lead to significant change required in existing design and build. Sometimes a small change required in one module of software may lead to significant changes required in other dependent modules. These factors result in the introduction of inadvertently introduced defects in the software.
- **Overloading of resources**—Due to intangibility and the abstract nature of any software product, it is very difficult to make a good schedule for software projects. This results in bad scheduling, which causes many software professionals to be overloaded. This results in more introductions of defects in the software. Most of the time software professionals find it difficult to meet deadlines, and this results in hasty and less careful work, resulting in more defects in the software.
- **Less skillful resources**—The software profession needs very skilled and experienced professionals. Many times less skilled professionals are recruited due to lack of skilled professionals in the market. This results in introduction of more defects in the software.
- **Unprofessional attitude**—Many times professionals on the project have attitude problems. They take their assignments lightly, play office politics, or try to shun their assigned work and off-load it to other members of the team. These tactics not only create problems in meeting deadlines for the project but also result in introduction of more defects in the software.
- **Poor documentation**—This is another primary source of defects in the software. In the case of smaller projects where some form of extreme programming or agile programming is followed, less documentation is acceptable because the project team is located at one place, and due to smaller software, complexity is less. But in the case of bigger projects where the team may be located at different sites and where software requirements are complex, good documentation is very important. Otherwise, the team runs a great risk of introducing more defects. Another aspect of poor documentation is difficulty in maintaining the software application after it goes into production.
- **Development tools**—In today's fast-paced business environment, project teams use many tools to increase their productivity. Some of these tools include visual tools, class libraries, compilers, scripting tools, RAD (rapid application development), integrated development studios, integration tools, and so on. Many times these tools introduce their own defects, and sometimes, due to their poor documentation, help in adding defects.